SEMINAR

RESOURCES

FOR S4

EDITIONS 1 & 2

PRINCIPLES OF OBSERVATION & EXPERIMENT

The "Good Experiment" Paradox

What is a Good Experiment?

1) A "Good Experiment" is one where
we always get the result we expect

These are the experiments every physics student does:
BUT of course
in principle such an experiment might fail!

2) A "Good Experiment" is one where
we don't know what to expect

This is what is ordinarily meant by an experiment:
BUT of course
there has to be some preconception about
what to observe and expect

In 1) reliable activity is the key:
emphasis on the "computational object"

In 2) interesting interpretation is the key:
emphasis on the "requirements analysis"

Paradox arises because the two kinds of experiment
on two different sides of an act-of-faith

Relating observation and experiment to system models

Any pattern of state change in a complex system
however initiated
can be modelled by recording
observations and synchronised change

1) In engineering terms, use to relate the behaviour o a complex system to experiments on its components

[Given the materials
could the Wright brothers have built an Airbus?]

2) Observation / experimentation of / on state gives more than synchronisation (cf parallel assignment state-transition model) : provides means to disentangle indivisibly linked state-change from independent state-change.

   Observation / experiment identifies agents

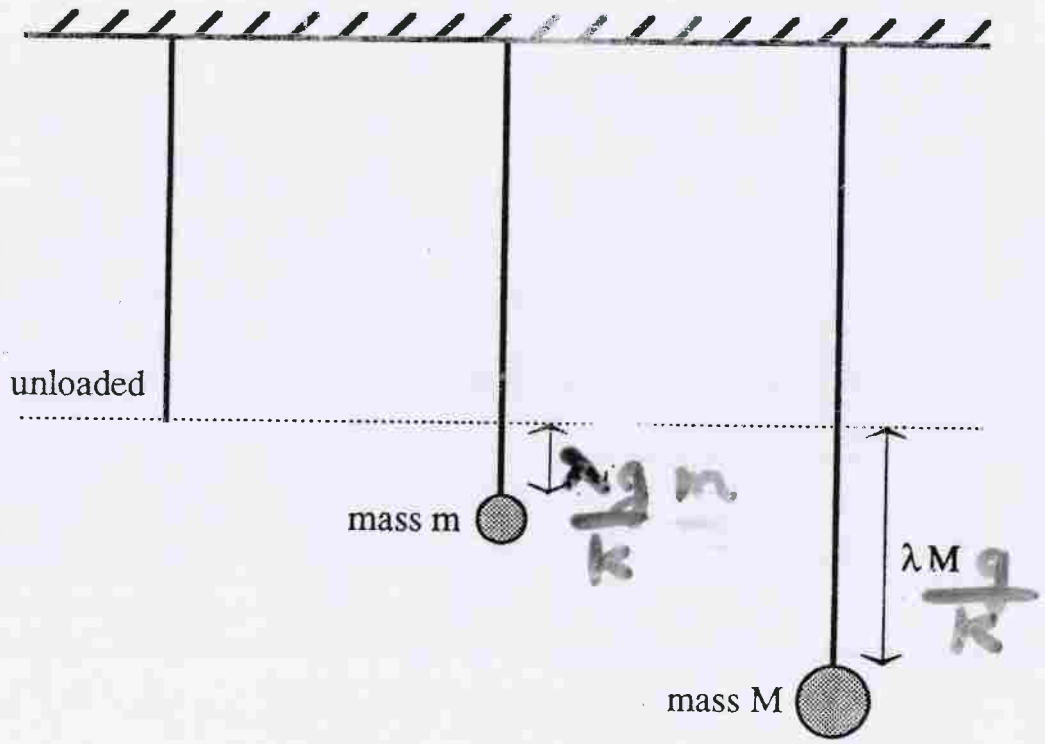
[Bertrand Russell quote]

3) Can simulate multi-agent interaction by multiple-action single agent : link from design to system simulation

   Designer plays the role of the agents: analysis by observation leads to *intelligibility* and *convenient redesign*

STRING EXTENSION IS PROPORTIONAL TO LOAD.

Hookes Law



λ = Young's Modulus

k = string constant

Expectation of observation

Expressed as correlation between observations

Functional relationship between one observation and another

can **compute** one observation from another: "that is"
can model the expected result by some other reliable expt

Mathematical perspective

- functional abstraction as
  "relation between observations
  made **in the same context**"

- "genuine" variables needed to represent observations
  i.e. can be evaluated in different contexts, have identity

Historically primary notions of **function** and **variable**

Bird and Wadler: Introduction to Functional Programming

".... every mathematician understands
that variables do *not* vary"

< Russian historian >

"In this mechanical picture of the world the essential, one might even say **definitive**, event was the concept of a law as a **dependence** between **variable quantities**."

Fyodor A Medvedev - a Russian historian

Scenes from the History of Real Functions,
Science Networks - Historical Studies Vol. 7,
Birhåuser-Verlag 1991

VISUALISATION IN SCIENTIFIC COMPUTING

cf. paper #025

"VISUALISATION ENTAILS MODELLING PHYSICAL PHENOMENA" (1

Analogies	between **physics**	and **computer science**
declarative	equational description of system behaviour mathematical theories	mathematical specification abstract computation functional / logic program
state-based + correlated to external world	heuristic models to aid interpretation of eqns	meaning of program how is connected to the real world
	Feynman: "completely unmathematical, imprecise and inexact nature of physical understanding"	Brian-Cantwell Smith: non-logicist view of computation: content relations aren't computed cf traditional PL semantics
modelling	describe state-transitions in physical systems	programming = modelling cf Simula philosophy

Issues for modelling in

	physics and	**computer science**
preconceptions	(classical physics) mechanical models	(current CS) object-oriented modelling
orthogonal views	modelling all aspects of physical phenomena inconsistent with single mechanical model	revision of requirements may => object redesign respecting locality of state-change problematic
faithfulness	how far is mechanism interpretable? how elaborate is the interpretation process?	how much of the object model refers to application? how does model describe how changes synchronise across object boundaries?

⇒ NEED A CONCEPT OF INTERPRETABLE PROGRAM STATE

SUMMARY

modern CS and classical physics have common themes and problems

- mathematically abstract aspect
- complementary interpretative state-based heuristics
- models that have similar qualities and limitations

Modern physics addresses the question

how is a physical phenomena correlated to the heuristic model?

	physics and	**computer science**
model environment and definition What is observable?	define observations identify variables whose values are monitored in describing behaviour	identify parameters to be monitored to specify operation of program in relation to its environment

⇒ NEED TO IDENTIFY WHAT STATE-CHANGING AGENTS CAN OBSERVE.

experimental context for the model How is it observed?	decide conventions for simultaneous observation	accuracy of observation granularity, how state-changes synchronise

⇒ NEED TO CONSIDER ACCURACY OF OBSERVATION, HOW ACTIONS AFFECT OBSERVATIONS

How is state changed?	change solely driven by experimentor: "what if?" e.g. Hooke's law	program state changed solely by user e.g. database, spreadsheet

⇒ NEED TO IDENTIFY PRIVILEGES OF AGENTS TO CHANGE STATE.

	change involving autonomous activity e.g. observation of planetary motion	program state is altered under program control and / or by independent agents cf reactive system

	physics	**and computer science**
How is state changed?	change solely driven by experimentor: "what if?" e.g. Hooke's law	program state changed solely by user e.g. database, spreadsheet
	change involving autonomous activity e.g. observation of planetary motion	program state is altered under program control and / or by independent agents cf reactive system
validation of model confirmation of experiment	correlation between simultaneous observation in change correctly predicted	program manipulates relevant parameters appropriately in relation to state of system

CHARACTERISTICS OF DEFINITIVE SCRIPTS

Features of use of definitions

definitive script = set of definitions

- order of definitions in text is immaterial
- each variable corresponds to a physical observation
- if a value of a variable is changed the values of other variables are automatically updated (cf spreadsheet)
- possible changes via redefinition might be: change mass of vehicle, adjust sampling speed on speed transducer, redesign speedometer, reconfigure dashboard display
- developing or embellishing design is also definition

Observations and Experiments in relation to Modelling

Indivisibly linked observations important in relation to many aspects of a model (e.g. speedometer model):

a) "semantics of the semantics" transformations of object

b) designer decides where to display, how to take account of negative speed

c) modes of propagation: mechanical linkage vs sampling

d) idealisation: speedo could be deemed to show actual speed

e) views / privileges : driver can't move / redesign speedo

f) non-computable content relations: exceeding speed limit? red warning light if cruise control has throttle at 0% and car is accelerating down a hill

Characteristic features of a definitive script 1

Script has a **state-based aspect**

- a variable in the script has a "procedural" nature: it has an *identity* and can assume different values

- a variable in the script designates a value that can be readily interpreted as an *observation*

- state interpretation of script defined wrt potential sequences of redefinitions and observations

In the state described by this script:
we observe the following values
+ subject to performing such and such redefinitions we shall be able to observe the following values

Characteristic features of a definitive script 2

Declarative perspective on scripts:

- the script has a declarative aspect: it expresses constraints on latent changes of state

- in computational terms, each defn represents computation that is taken for granted

y = f(x) in the script S means:
It is a feature of my computer that in the state defined by the script S when you change the value of x the value of y also changes simultaneously according to the formula y=f(x)

Interpret script
ia instantaneous observation of state
The processes that maintain consistency are hidden

Characteristic features of a definitive script 3

- a script is good for describing a state-transition correlated with observation in experiment

You can confirm that my script does / doesn't accurately respresent observed rels between values in change

Corollary: there is an objective criterion for evaluating a script as a representation of observed external rels

- until we specify the permissible redefns a script does not define a state-transition model at all

If there's no restriction on what can be redefined I can transform any script into any other script

Actually need to specify restrictions to establish exact correspondence between script and expt'l observation

- a script plus a protocol for redefn is required to specify a behaviour

Typical role for definitive scripts

Use of the scripts oriented towards activities:
requirements analysis -> specification
experiment -> theory
conceptual design -> specification for manufacture

Requirements / experiment involves knowledge of specific properties in isolation

These restrict redefns on scripts we use to model

Only when we've decided what full range of permissible state-changes that's appropriate do we formally specify

At this stage first have a concept of gloabl behaviour about which we'd like to reason, prove properties etc

The greatest common divisor folk-dance routine

To calculate GCD(m,n), take m woman and n men
NB m and n must be positive

Match up men and women in pairs
until no more pairs can be formed
If everybody has a partner
stop the dance and count the number of pairs
[This'll be GCD(m,n)]

Otherwise
there's either a spare man or a spare woman
NB of course there may be more than one!
If there's a spare man:
send the men with partners out of the room
NB without their partners
If there's a spare woman:
send the women with partners out of the room
NB without their partners

Repeat the dance, forming new pairs etc

Moral

Computation does not require computers
Computer Science is not fundamentally about
machines / machine code

Definitive scripts good for
reactive systems requirement

- program model is defined by **observations** of reliable state-changing devices
- **observations** determine the communication of state-changes between agents
- the process of interpretation involves matching of experimental observations

Spreadsheet principles enable us to specify observations e.g. represent objects in way faithful to observation

Express indivisibility: cf OOP

1st & 2nd FACTOR

B. C. SMITH

2 LESSONS OF LOGIC

Spreadsheet principles good wrt Smith's thesis

Can express behaviour of the content relation e.g. doodling vs signing away my house

This is not a part of the computation to be carried out by devices: no **execution** aspect

context dependence of definitions

procedural nature of underlying intuition
observation (= identity + change)
not a mathematical variable

more prescriptive modelling

Semantics of a geometric object reflected in its defn cf MacDraw picture and script definition

MOD 1

STATE

```
openshape cabinet
within cabinet {
    int       width, length
    point     NW, NE, SW, SE
    line      N, S, E, W

    N = [NW, NE]
    S = [SW, SE]
    E = [NE, SE]
    W = [NW, SW]

    width, length = 300, 300

    SW = {100, 200}
    SE = SW + {width, 0}
    NW = SW + {0, length}
    NE = NW + {width, 0}

    openshape drawer
    within drawer {
        bool      open
        int       length
        line      N, S, E, W

        length = if open then ~/length else 0
        open = true

        N = [~/NW + {0, length},  ~/NE + {0, length}]
        S = [~/NW, ~/NE]
        W = [~/NW + {0, length}, ~/NW]
        E = [~/NE + {0, length} , ~/NE]
    }
}

protocol {
    open -> open = false
    ! open ∧ ! locked -> open = true
    locked -> locked = false
    ! open ∧ ! locked -> locked = true
}
```

POSSIBLE TRANSITIONS

C.F. USE OF STATE IN CONVENTIONAL PROCEDURAL GRAPHICS PACKAGE.

MOD 2

STATE

```
openshape led
within led {
    int     digit
    point   p1, p2, p3, p4, p5, p6
    line    L1, L2, L3, L4, L5, L6, L7
    bool    on1, on2, on3, on4, on5, on6, on7

    digit = 8

    p1 = {100, 800}
    p2 = {100, 500}
    p3 = ...

    on1 = digit != 1 ∧ digit != 4
    on2 = digit != 0 ∧ digit != 1 ∧ digit != 7
    on3 = ...

    l1 = if on1 then [p1, p4] else [p1, p1]
    l2 = if on2 then [p2, p5] else [p2, p2]
    l3 = ...
}

protocol {
    true -> digit = | digit | + 1
    true -> digit = 0
}
```

POSSIBLE TRANSFORMATIONS

WHAT DOES THIS DEPICT?